7 Zusammenfassung und Ausblick

Die Entwicklung großer Anwendungssysteme sowohl im industriellen als auch im wissenschaftlichen Bereich stellt stets eine enorme Herausforderung dar. Diese Arbeit leistet einen Beitrag, diese Herausforderung im konstruktiven Bereich besser bewältigen zu können. Dazu wurden Ergebnisse aus den Gebieten Modellbildung im Softwareentwicklungsprozeß, Semiotik, Organisationstheorie, Softwarearchitekturen und objektorientierte Anwendungsentwicklung zueinander in Beziehung gesetzt. Durch die inhaltliche Verknüpfung dieser Bereiche weist der vorgestellte Ansatz insbesondere die folgenden Vorzüge auf:

- Der *objektorientierte Schichtenarchitekturstil organisiert*, im Gegensatz zu herkömmlichen Architekturstilen (vgl. [SG96]), sowohl das *fachliche* als auch das *softwaretechnische Modell*. Die Notwendigkeit hierzu ergab sich aus den Erkenntnissen, die zum einen bei der Aufarbeitung der Modellbildung im Softwareentwicklungsprozeß und zum anderen aus der Diskussion von Ergebnissen der Semiotik gewonnen wurden.
- Die *Makrostrukturen* des fachlichen Modells beruhen dabei auf der *Organisationsstruktur* des Anwendungsbereichs, die des softwaretechnischen Modells auf denen des fachlichen Modells und der verwendeten Technologie. Diese *Strukturähnlichkeit* zwischen dem Anwendungsbereich, dem fachlichem und softwaretechnischem Modell sowie der verwendeten Technologie ermöglicht, ebenso wie bei Mikrostrukturen, *eine bruchlose Umsetzung der fachlichen Konzepte* des Anwendungsbereichs bis hin in das softwaretechnische Modell. Der erarbeitete Schichtenarchitekturstil kann daher zur Konstruktion von objektorientierten Softwaresystemen verwendet werden, deren Anwendungsbereich nach dem Produkt- oder Prozeßprinzip organisiert ist. Der homologe Aufbau der Modelle und Kontexte erleichtert zudem sowohl die *Etablierung* als auch die *Reetablierung* von *Codes* im Softwareentwicklungsprozeß.
- Da die Makrostrukturen des fachlichen und softwaretechnischen Modells von den Organisationsstrukturen des Anwendungsbereichs abgeleitet wurden, konnte zudem eine *Zerlegung* von Teilen der beiden Modelle *in getrennt voneinander modellierbare Produktbereiche* erzielt werden. Insbesondere diese Zerlegung schafft die Grundlage für eine evolutionäre Entwicklung großer Anwendungssysteme.
- Die Schichtenarchitektur im softwaretechnischen Modell nimmt eine *saubere Trennung zwischen der verwendeten Technik und dem Modell der verwendeten Technik* in Form der beiden Schichten Systembasis und Technologie vor. Dies ermöglicht es, Softwaresysteme, die nach diesem Architekturstil konstruiert worden sind, auf unterschiedliche Systemplattformen zu portieren. Außerdem ist die einheitliche Verwendung eines Technikmodells innerhalb einer Familie von Anwendungssystemen gewährleistet.
- Eine *konstruktiv saubere Realisierung* der vorgestellten Architekturstile, die vor allem eine einfache Evolution der Schichten und der darin entwickelten Komponenten ermöglicht, wird durch die beiden Entwurfsmuster Produkthändler und Rollen gewährleistet.
- Die *Konfiguration von Anwendungssystemen* auf der Basis von Komponenten des präsentierten Schichtenarchitekturstils läßt sich mittels der Händlerkonfigurationsskripten realisieren.

Die nachfolgende Argumentationskette faßt nun die inhaltlichen Schwerpunkte und die daraus resultierenden Ergebnisse meiner Arbeit abschließend zusammen und gibt gleichzeitig einen Ausblick auf weitere Forschungsarbeiten.

Während des Softwareentwicklungsprozesses konstruieren die Mitarbeiter eines Projektteams eine Reihe von Modellen, die sich gegenseitig beeinflussen und durch äußere Kontexte geprägt werden. Als prägende Kontexte wurden in Kapitel 2 der Anwendungsbereich, die Technologie sowie die verwendete Technik herausgearbeitet. Der Anwendungsbereich bildet im Softwareentwicklungsprozeß die fachliche Ausgangslage für das zu konstruierende fachliche Modell. Daneben wird es von den eingesetzten Leitbildern und Entwurfsmetaphern geprägt. Das fachliche Modell stellt nun seinerseits wieder die Grundlage für das softwaretechnische Modell dar. Aufgabe des softwaretechnischen Modells ist dabei die Einbettung des fachlichen Modells in einen gegebenen technikgeprägten Kontext.

Diesen Kontext habe ich in meiner Arbeit in die Bereiche verwendete Technik und Modell der verwendeten Technik unterteilt. Die verwendete Technik umfaßt dabei die zur Konstruktion eine Anwendungssystems benötigte Technik sowie die eingesetzte Technik, um das Anwendungssystem so zu gestalten, daß es den gestellten Anforderungen bestmöglich genügen kann. Neben diesem rein auf die Verwendung von Schnittstellenoperationen fokussierten Einsatz der Technik benutzt ein Softwareentwickler Technik auch in Form eines Modells, das er über die zu verwendete Technik besitzt. Dieses Modell repräsentiert das Wissen des Softwareentwicklers oder eines Projektteams darüber, wie die Technik zur Konstruktion des softwaretechnischen Modells zu verwenden ist.

Neben der verwendeten Technik und dem korrespondierenden Modell beeinflussen insbesondere Entwurfsmuster und Softwarearchitekturen die Konstruktion des softwaretechnischen Modells. Entwurfsmuster beschreiben dabei auf Mikroebene, wie ein bestimmtes, kontextbezogenes Entwurfsproblem gelöst werden kann. Eine Softwarearchitektur organisiert und strukturiert das softwaretechnische Modell als eine Menge interagierender Komponenten und Verbindungsstücke. Sie beschreibt demnach den Gesamtaufbau eines Anwendungssystems auf Makroebene. Entwurfsmuster und Softwarearchitekturen stellen somit sich ergänzende Strukturierungsmittel dar, die ein Anwendungssystem auf unterschiedlicher Granularitätsebene organisieren.

Die Gesamtheit der drei Bereiche Leitbilder und Entwurfsmetaphern, Entwurfsmuster und Softwarearchitekturen sowie Modell der verwendeten Technik habe ich unter dem Begriff Technologie zusammengefaßt, da sie die technischen Kenntnisse, Fähigkeiten und Möglichkeiten repräsentieren, die die Konstruktion eines Anwendungssystems beeinflussen. Das so entstandene Prozeßmodell der objektorientierten Softwareentwicklung wurde dann in den anschließenden Kapiteln als Grundlage für die zu diskutierenden Fragestellungen verwendet.

Neben der Grundlage, die das erste Kapitel für meine Arbeit bildet, sehe ich als relevantes Ergebnis insbesondere die von mir vorgestellte Trennung in Modelle und ihre beeinflussenden Kontexte sowie die Unterscheidung in verwendete Technik und Modell der verwendeten Technik.

In Kapitel 3 habe ich zunächst untersucht, welche strukturellen Abhängigkeiten zwischen den erstellten Modellen und ihren prägenden Kontexten vorteilhaft sind. Motiviert wurde diese Fragestellung anhand der Tatsache, daß die mit der objektorientierten Anwendungsentwicklung verbundene bruchlose Modellierung von vielen Autoren als einer der entscheidenden Vorteile gegenüber herkömmlichen Analyse- und Designmethoden gesehen wird. Dieser Vorteil beruht auf der Möglichkeit, die Mikrostrukturen des Anwendungsbereichs (z.B. eine Begriffshierarchie) unter Einsatz einer passenden Technologie (z.B. Objektorientierung) homolog in das fachliche und softwaretechnische Modell zu übertragen (z.B. in eine Klassenhierarchie). Zur Beantwortung der Fragestellung nach strukturellen Abhängigkeiten habe ich Aussagen aus der Semiotik auf das vorgestellte Prozeßmodell der objektorientierten Softwareentwicklung übertragen.

Ausgehend von der erarbeiteten Erkenntnis, daß dem Prozeßmodell ein Signifikationssystem zugrunde liegt, wurde der Zusammenhang zwischen den Modellen und ihren beeinflussenden Kontexten mittels konnotativer und denotativer Semiotiken beschrieben. Die Semiotiken beruhen dabei auf der Etablierung der nachfolgend beschriebenen Codes:

- das Ausdruckssystem »softwaretechnisches Modell« wird durch einen Code mit dem Inhaltssystem »verwendete Technologie« korreliert;
- das Ausdruckssystem »softwaretechnisches Modell« wird durch einen weiteren Code mit dem Inhaltssystem »fachliches Modell« verbunden;
- das Ausdruckssystem »fachliches Modell« wird durch einen Code mit dem Inhaltssystem »Anwendungsbereich« korreliert;
- das Ausdruckssystem »fachliches Modell« wird auch durch einen Code mit dem Inhaltssystem »Leitbild und Entwurfsmetaphern« verbunden;
- das Ausdruckssystem »fachliches Modell« wird zudem durch einen Code mit dem Inhaltssystem »Softwarearchitektur« korreliert.

Für die Etablierung dieser Codes stellt die Struktur von Ausdrucks- und Inhaltssystem eine wichtige Eigenschaft dar. Die erstellten Modelle sollten daher homolog zu den beeinflussenden Kontexten sein. Eine Softwarearchitektur muß folglich immer auf Basis einer Technologie und auf Basis einer Klasse strukturell verwandter Anwendungsbereiche definiert werden. So lassen sich sowohl die im Anwendungsbereich vorhandenen Mikrostrukturen als auch die Makrostrukturen bruchlos zur verwendeten Technologie in das fachliche und softwaretechnische Modell übertragen.

Da in den weiteren Kapitel insbesondere Fragen in Bezug auf Mikro- und Makrostrukturen im Anwendungsbereich und der verwendeten Technologie diskutiert wurden, habe ich weiterhin die mit dem Begriff Struktur verbundenen Merkmale thematisiert. Diese wurden wie folgt angegeben: eine Struktur beschreibt den inneren Aufbau, das Gefüge eines Systems. Sie kann verwendet werden, um unterschiedliche Systeme von einem einheitlichen Gesichtspunkt aus zu standardisieren oder um in Systemen uns vertraute Eigenschaften zu identifizieren.

Offen geblieben, und damit potentiell Gegenstand weiterer Forschungsarbeit, ist in dem Kapitel 3 die Frage, welche Ergebnisse aus der Semiotik der Kommunikation ebenfalls auf den Softwareentwicklungsprozeß übertragbar sind.

Die in Kapitel 3 gestellte Forderung, daß die erstellten Modelle homolog zu ihren beeinflussenden Kontexten zu organisieren sind, implizierte eine Diskussion über Unternehmensorganisationen. Denn anwendungsfachliche Makrostrukturen, die als Grundlage für eine Softwarearchitektur dienen können, sind bei Softwaresystemen für den Dienstleistungsbereich in Organisationen oder in Bereichen einer Organisation zu suchen. Zudem verfügen die Strukturen eines Unternehmens über eine ausreichend hohe Lebensdauer, um als Basis für eine Softwarearchitektur zu dienen.

Die Diskussion der aus der Organisationstheorie bekannten Organisationsmodelle und ihre Gegenüberstellung mit einer rein technologisch motivierten Struktur, die nach den Modularisierungsprinzipien minimierte Kopplung und maximierte Kohäsion gebildet wurde, hat gezeigt, daß sich im Kontext objektorientierter Softwareentwicklung Organisationsstrukturen, die nach dem Produkt- oder Prozeßprinzip gebildet sind, als Grundlage für eine Softwarearchitektur eignen. Bei Unternehmen, die nach dem Prozeßprinzip strukturiert sind, müssen allerdings die Prozesse überwiegend an den Produkten orientiert sein.

Die Übertragung der Ergebnisse aus der Organisationstheorie auf das fachliche Modell hat dort zu den Bereichen Einsatzkontexte und Produktbereiche geführt. In einer daran anschließenden Diskussion über den Inhalt von Produktbereichen habe ich herausgearbeitet, daß zwischen verschiedenen Produktbereichen typischerweise Überschneidungen bestehen, die sich wie folgt charakterisieren lassen: (1) Begriffe werden über die einzelnen Produktbereiche hinweg gemeinsam verwendet, (2) zu Begriffen unterschiedlicher Produktbereiche ist eine gemeinsame Generalisierung vorhanden oder (3) mit Gegenständen, die anscheinend getrennt voneinander in den verschiedenen Produktbereichen existieren, ist eine gemeinsame fachliche Identität verbunden. Die Existenz dieser Überschneidungsarten und die Tatsache, daß die Konstruktion von fachlich integrierten Anwendungssystemen eine gemeinsame fachliche Basis voraussetzt, hat zu der Definition des Gegenstandsbereichs geführt. Dieser umfaßt somit als Modellierungsbereich die fachlichen Kernkonzepte der verschiedenen Produktbereiche.

Der Gegenstandsbereich repräsentiert demnach ein für alle Produktbereiche gemeinsames Modell. Um nicht mit denselben Problemen konfrontiert zu sein, die mit der Entwicklung eines unternehmensweiten Modells (Objekt- oder Datenmodell) verbunden sind, habe ich notwendige Eigenschaften des Gegenstandsbereichs aufgeführt sowie Vorgehensweisen und Konstruktionsprinzipien für seine technische Realisierung beschrieben. Dabei wurde insbesondere das Offen-Geschlossen-Prinzip als ein wichtiges Konstruktionsprinzip identifiziert.

Die von mir in Kapitel 4 vorgeschlagene Methode, die Makrostrukturen für das fachliche und softwaretechnische Modell von den Makrostrukturen des Anwendungsbereichs abzuleiten, sehe ich vor allem für Anwendungssysteme, die für den Dienstleistungssektor entwickelt werden, als wissenschaftlich relevant an. Denn sie ermöglicht es, nicht nur die Mikrostrukturen sondern auch die Makrostrukturen bruchlos bis in das softwaretechnische Modell zu

übertragen. Daher ergibt sich als weitere Forschungsfrage, inwieweit Organisationsstrukturen, die nach dem Verrichtungsprinzip gebildet worden sind, als Grundlage einer Softwarearchitektur für Anwendungssysteme dienen könnten, die nach der klassischen Methode der strukturierten Analyse und des strukturierten Entwurfs entwickelt werden sollen. Denn auch zwischen dem für die Organisation des Unternehmens und dem für die Architektur des Softwaresystems verwendeten Strukturierungsprinzip ist eine Strukturähnlichkeit unverkennbar. Ebenfalls unberücksichtigt blieb die Fragestellung, ob sich das Modell aus Gegenstandsbereich und Produktbereich wieder rekursiv auf einen Produktbereich anwenden läßt. Dies scheint eine konsequente und logische „Wiederverwendung“ des vorgestellten Musters zu sein. Die rekursive Anwendung des Modells wurde von mir aber nicht weiter beleuchtet, da hierzu noch keine Erfahrungen im Bereich großer objektorientierter Systeme existieren.

Ziel des Kapitel 5 war, die in Kapitel 4 erarbeitete Struktur des fachlichen Modells auf ein objektorientiert geprägtes softwaretechnisches Modell zu übertragen. Die Beschreibung der Makrostruktur des softwaretechnischen Modells erfolgte dabei in Form zweier Architekturstile. Ich habe mich in diesem Kapitel zuerst konzeptionell mit den Eigenschaften von Architekturstilen und Architekturbeschreibungssprachen, den Verwendungsformen von Rahmenwerken sowie den Merkmalen von Schichtenarchitekturen auseinandergesetzt. Aufbauend auf diesen konzeptionellen Erörterungen habe ich Beispiele objektorientierter Schichtenarchitekturen diskutiert. Dabei hat sich gezeigt, daß die Drei-Schichten-Architektur zwar eine geeignete Grundlage für eine Systemarchitektur darstellt, sich aber nicht als Basis einer Softwarearchitektur für große objektorientierte Anwendungssysteme eignet. Logische Schichtenarchitekturen, die Komponenten in schnittstellenlose Schichten gruppieren, wie dies etwa auch bei ET++ der Fall ist, eignen sich hingegen bestens. Insbesondere ermöglicht es diese Form der Schichtenarchitektur mittels Vererbung und Polymorphie, das Offen-Geschlossen-Prinzip umzusetzen. Dies bedeutet, daß sich Komponenten tieferliegender Schichten in höherliegenden Schichten einfach erweitern lassen. Eben diese Eigenschaft wurde zuvor als das Kernkonstruktionsprinzip herausgearbeitet, um die Nachteile zu vermeiden, die mit der Entwicklung von unternehmensweiten Objekt- bzw. Datenmodellen verbunden sind.

Ausgehend von dieser Diskussion, den Ergebnissen aus Kapitel 4 sowie der Unterscheidung in verwendete Technik und Technologie, die ich in Kapitel 2 vorgestellt habe, wurde ein objektorientierter Schichtenarchitekturstil definiert. Die Unterscheidung in verwendete Technik und Technologie hat dabei zu den beiden Schichten Systembasis und Technologie geführt. Die Schichten Gegenstandsbereich, Produktbereich und Einsatzkontext wurden strukturerhaltend vom fachlichen auf das softwaretechnische Modell übertragen.

Die durch die logischen Schichten des objektorientierten Schichtenarchitekturstils gruppierten Komponenten und deren Zusammenspiel habe ich mittels eines rahmenwerkbasierten Architekturstils beschrieben. Dabei stellen die Rahmenwerke der Technologieschicht die technische, die Rahmenwerke der Gegenstandsbereichsschicht die fachliche Kombinierbarkeit der in den Produktbereichen und Einsatzkontexten entwickelten Komponenten sicher. Zudem hat die Kombination der beiden Architekturstile eine detailliertere Angabe von Regeln, die das Zusammenspiel von Komponenten in unterschiedlichen Schichten beschreiben, ermöglicht.

Den Abschluß des Kapitels 5 bildet die technische Realisierung der beiden vorgestellten Architekturstile. Dabei habe ich gezeigt, daß dynamische Bibliotheken die mit einem Rahmenwerk verbundenen Ansprüche, Analysen, Entwurfsentscheidungen, Architekturen und Implementierungen wiederzuverwenden, sinnvoll umsetzen. Allerdings führten technische Argumente, wie unnötige Übersetzungen oder eine verbesserte Konfektionierung der Rahmenwerke, zu einer Trennung von Rahmenwerk- und Bibliotheksstruktur. Diese Trennung wurde ebenfalls auf die Komponente Klassenbibliothek übertragen.

Als Grundkonzept zur Lösung dieser technischen Probleme habe ich die Aufteilung von Rahmenwerken und Klassenbibliotheken in einen Konzeptteil und mehrere Realisierungsteile vorgeschlagen. Ausgehend von dieser Unterteilung werden Konzeptteile verschiedener Rahmenwerke und Klassenbibliotheken unter Berücksichtigung von Modularisierungsprinzipien zu dynamischen Bibliotheken zusammengefaßt. Die Bündelung der Implementierungsteile erfolgt analog. Die dadurch entstandene bibliotheksorientierte Sicht auf das aus Schichten, Klassenbibliotheken und Rahmenwerken bestehende softwaretechnische Modell wird mittels virtueller Dateisysteme oder symbolischer Verweise realisiert.

Die Mikrostrukturen eines Anwendungssystems werden durch die eingesetzte Programmiersprache, die Makrostruktur durch die gewählte Softwarearchitektur festgelegt. Die beiden Strukturen eines Anwendungssystems sind aber heute nur unzureichend miteinander gekoppelt. Eine Verbindung existiert i.d.R. nur in Form von Dokumentation. Wird als Softwarearchitektur beispielsweise eine undurchlässige Schichtenarchitektur verwendet, so kann heute auf Mikroebene eine Klasse trotzdem Klassen verwenden, die nicht nur in der angrenzenden, tieferliegenden Schicht enthalten sind. Ein Bruch zwischen der gemäß der gewählten Softwarearchitektur erwünschten Makrostruktur und der durch die implementierte Mikrostruktur tatsächlich realisierten Makrostruktur führt zum Verlust der Gesamtarchitektur des Anwendungssystems (vgl. [GAO95]). Es sind daher neben den Regeln, die für einen Architekturstil festlegen, wie dessen Komponenten und Verbindungsstücke miteinander kombiniert werden dürfen (vgl. [SG96]), Werkzeuge in Softwareentwicklungsumgebungen und Erweiterungen von Programmiersprachen erforderlich, mit denen sich die Makrostruktur eines Anwendungssystems beschreiben und so bei der Übersetzung gegen die realisierten Strukturen prüfen läßt. Diese Fragestellung eröffnet meines Erachtens einen für die Zukunft der Softwaretechnik interessanten Forschungsbereich.

Offengeblieben ist weiterhin die Problematik, welche Auswirkungen der Einsatz unterschiedlicher technologischer Konzepte auf die Schichtenarchitektur hat. Wenn nämlich Rahmenwerke im Gegenstandsbereich und in den Produktbereichen auf unterschiedlicher technologischer Basis konstruiert werden, dann können diese in den Einsatzkontexten nicht mehr einfach zu Anwendungssystemen kombiniert werden. Diese Fragestellung ist damit ebenfalls Gegenstand weiterer Forschungsarbeit.

In Kapitel 5 habe ich deutlich gemacht, daß sich bestimmte Entwurfsmuster als schichtenübergreifende Verbindungsstücke zwischen Rahmenwerken einsetzen lassen. Daher sind in Kapitel 6 die meines Erachtens wichtigen Konzepte Produkthändler und Rollen unter diesem Gesichtspunkt diskutiert worden.

Das Konzept des Händlers setzt sich mit dem Sachverhalt auseinander, daß ein Klient, wenn er eine gewünschte Dienstleistung von einem Objekt in Anspruch nehmen will, zuvor eine Referenz auf das Objekt besitzen muß. Diese kann ein Klient entweder dadurch erhalten, daß ihm das Objekt zuvor durch eine passende Methode übergeben wurde, oder indem er mittels Aufruf der `new` Operation ein entsprechendes Objekt erzeugt. Die Instanziierung des Objekts durch den Aufruf der `new` Operation ist aber unter folgender Rahmenbedingung problematisch: wird die Dienstleistung von einem Objekt erbracht, dessen Klasse in einer höherliegenden Schicht oder in einem benachbarten Produktbereich enthalten ist, so kann dieses auch dann nicht erzeugt werden, wenn der spätere Zugriff auf das Objekt nur unter der Schnittstelle seiner aufgeschobenen Oberklasse erfolgt. Denn der Aufruf der `new` Operation erzeugt eine Abhängigkeit zu einer höherliegenden Schicht oder zu einem benachbarten Produktbereich.

Wie eine ausführliche Diskussion der in [GOF95] vorgestellten Erzeugungsmuster gezeigt hat, lösen diese die aufgezeigte Problematik im Zusammenhang mit einer Schichtenarchitektur nur unzureichend. Ich habe daher das Entwurfsmuster des Produkthändlers präsentiert, bei dem ein Klient die Vermittlung (Instanziierung bzw. Wiederverwendung) eines bestimmten Objekts an einen Händler delegiert. Der Klient beschreibt dabei die Dienstleistung, die von dem Objekt erbracht werden soll, in Form einer Spezifikation. Der Zugriff auf das Objekt erfolgt dann lediglich unter Verwendung von Schnittstellen, die in der Schicht des Klienten oder in einer tieferliegenden Schichten deklariert sind. Folglich können Klienten durch den Einsatz eines Produkthändlers sowohl von der Objekterzeugung als auch vom Typ und somit von der Existenz konkreter Produktklassen vollständig entkoppelt werden.

Seine Konfiguration verwaltet ein Händler in Form von Zuordnungen. Diese Form der Implementierung verbunden mit dem Ansatz, den Aufbau eines Anwendungssystems unter Verwendung von Konfigurationsskripten zu beschreiben, führt zu den folgenden Eigenschaften von Softwaresystemen:

- Die Konfiguration eines Händlers läßt sich dynamisch anpassen. Das Verhalten eines Anwendungssystems kann somit zur Laufzeit einfach verändert werden.
- Anwendungssysteme lassen sich durch Konfigurationsskripten auf der Basis bereits bestehender Rahmenwerke konfektionieren. Die Summe aller Händlerkonfigurationen beschreibt somit den Aufbau des Gesamtsystems auf einer abstrakten Ebene.
- Die Konfiguration eines Anwendungssystems bzw. Rahmenwerks kann verändert werden, ohne daß dazu ein Rahmenwerk geöffnet werden muß. Dieses Merkmal ermöglicht insbesondere eine einfache Evolution von Rahmenwerken und Anwendungssystemen.

Bereits bei der Diskussion des Gegenstandsbereichs in Kapitel 4 wurde deutlich, daß das Konzept der Vererbung nicht ausreicht, um Begriffe des Anwendungsbereichs, die aus dem Blickwinkel verschiedener Produktbereiche unterschiedlich modelliert werden, softwaretechnisch befriedigend umzusetzen. Um diese Problematik im Kontext des vorgestellten Schichtenarchitekturstils sauber lösen zu können, habe ich das Konzept der Rolle eingeführt. Rollen ermöglichen es, die aus der Sicht eines Produktbereichs fachlich zusammengehörigen Zustände und Umgangsformen eines Konzepts in einer eigenen Klasse zu modellieren. Dazu wird ein

Konzept in ein Kern- und mehrere Rollenkonzepte aufgeteilt. Das Kernkonzept ist typischerweise Bestandteil des Gegenstandsbereichs, die einzelnen Rollenkonzepte sind Teil der verschiedenen Produktbereiche. Rollenkonzepte ermöglichen es somit, Kernkonzepte des Gegenstandsbereichs zu erweitern, ohne diesen dazu öffnen zu müssen.

Die objektorientierte Umsetzung von Kern- und Rollenkonzept erfolgt durch korrespondierende Kern- und Rollenobjekte. Um dennoch die fachliche Identität des durch Kern- und Rollenobjekt realisierten Konzepts zu wahren (Kern- und Rollenobjekte stellen eigene technische Identitäten dar) habe ich auf der Grundlage der einschlägigen Literatur die Begriffe Subjekt und Subjektidentität eingeführt. Anschließend habe ich mit dem Rollenmuster die technische Realisierung der Konzepte Kernobjekt, Rollenobjekt und Subjekt in einer typisierten objektorientierten Programmiersprache beschrieben.

Wie die Zusammenfassung zeigt, hat sich die Arbeit mit vielen scheinbar unabhängig voneinander existierenden Bereichen der Softwareentwicklung beschäftigt. Ziel dabei war es, diese Bereiche so miteinander in Relation zu setzen, daß dadurch die Entwicklung großer rahmenwerkbasierter Anwendungssysteme vereinfacht wird. Neben dem konzeptionellen Beitrag, den diese Arbeit hierbei geleistet hat, sind aus ihr auch eine Reihe von softwaretechnisch interessanten Fragestellungen für zukünftige Forschungsarbeiten hervorgegangen.